



ARL-TR-7546 • DEC 2015



US Army Research Laboratory

Interference Cancellation System Design Using GNU Radio

by Jan Paolo Acosta

Approved for public release; distribution unlimited.

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.



Interference Cancellation System Design Using GNU Radio

by Jan Paolo Acosta

Sensors and Electron Devices Directorate, ARL

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
<p>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p>					
1. REPORT DATE (DD-MM-YYYY) December 2015		2. REPORT TYPE Final		3. DATES COVERED (From - To)	
4. TITLE AND SUBTITLE Interference Cancellation System Design Using GNU Radio				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Jan Paolo Acosta				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Research Laboratory ATTN: RDRL SER U 2800 Powder Mill Road Adelphi, MD 20783-1138				8. PERFORMING ORGANIZATION REPORT NUMBER ARL-TR-7546	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT This report contains information on how to design an interference cancellation system using Ettus software-defined radios and the GNU Radio software development kit.					
15. SUBJECT TERMS interference cancellation, software-defined radio					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 24	19a. NAME OF RESPONSIBLE PERSON Jan Paolo Acosta
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (Include area code) 301-394-1530

Contents

List of Figures	iv
1. Introduction	1
2. Development	1
2.1 Hardware Platform	1
2.2 Software Platform	3
2.3 Adaptive Filter Algorithm	3
3 Performance	4
3.1 Test Setup	4
3.2 Test Parameters	5
3.3 Test Results	5
4. Summary and Conclusion	8
5. References	9
Appendix. GNURadio C++ Code	11
List of Symbols, Abbreviations, and Acronyms	17
Distribution List	18

List of Figures

Fig. 1	Basic interference cancellation scheme	1
Fig. 2	Ettus USRP B210.....	2
Fig. 3	Interference cancellation test setup	4
Fig. 4	Adaptive filter length of 1	6
Fig. 5	Adaptive filter length of 2.....	7
Fig. 6	Adaptive filter length of 3.....	7
Fig. 7	Adaptive filter length of 5.....	8

1. Introduction

Interference cancellation is a technique mainly used to reduce co-channel interference. The basic scheme for interference cancellation is shown in Fig. 1. This figure shows that a desired signal is corrupted by interference. The primary signal contains both the desired signal $s(n)$ and the noise component $n_2(n)$. The noise component is a function of the reference passing through a channel with an impulse response $w(n)$. A reference of the noise component $n_1(n)$ is then fed to a finite impulse response (FIR) filter that adaptively tunes using the difference between the adaptive filter output and the primary signal. The impulse response of the adaptive filter will try to match the impulse response $w(n)$. If the output of the adaptive filter is equal to the noise component of the primary signal, then the error of the adaptive filter is equal to the desired signal $s(n)$.

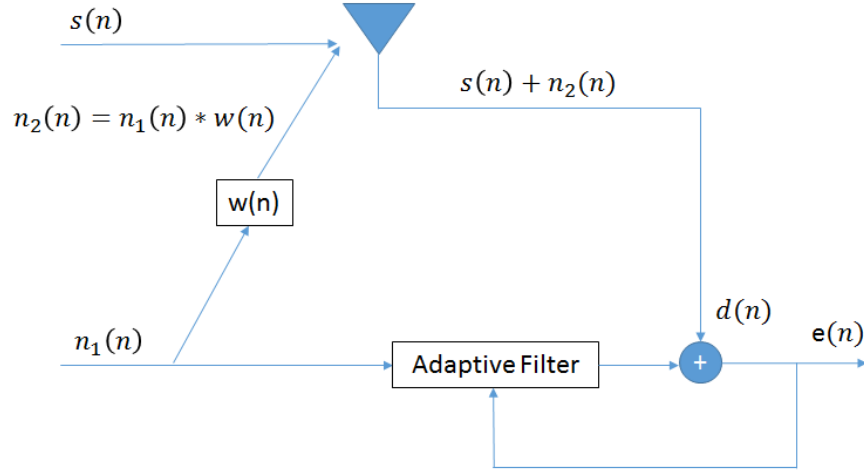


Fig. 1 Basic interference cancellation scheme

2. Development

2.1 Hardware Platform

To reduce development time, commercial-off-the-shelf (COTS) radios sold by National Instruments subsidiary Ettus Research were used as a development platform. Interference cancellers at minimum require 2 inputs and 1 output. Ettus Research sells a number of software-defined radios (SDRs) that are capable of meeting this requirement. These SDRs are also used with the GNU Radio software development kit (SDK), an open source platform that is available for free in Linux.

The SDR chosen for testing is the universal software radio peripheral (USRP) B210 and is shown in Fig. 2.



Fig. 2 Ettus USRP B210

According to Ettus Research, the B210 has the following specifications:

- 1) Fully integrated, 2-channel device with continuous radio frequency (RF) coverage from 70–6 GHz.
- 2) Full duplex, multiple input multiple output (MIMO) (2 transmit and 2 receive) operation with up to 56 MHz of real-time bandwidth (61.44 MS/s quadrature)
- 3) USB 3.0 connectivity
- 4) GNU Radio and OpenBTS support
- 5) Open and reconfigurable Spartan 6 XC6SLX150 field-programmable gate array (FPGA)
- 6) Analog-to-digital converter (ADC) and digital-to-analog converter (DAC) resolution of 12 bits.
- 7) Receive noise figure of less than 8 dB

Based on the USRP B210 specifications, the theoretical interference cancellation performance achievable is $6\text{ dB} \times 12\text{ bits} = 72\text{ dB}$.

2.2 Software Platform

Signal processing algorithms were developed using the GNU Radio. GNU Radio is a free and open source SDK that provides signal processing blocks to implement software radios. It is widely used in hobbyist, academic, and commercial environments to support both wireless communications research and real-world radio systems. Using the GNU Radio simplifies the development process and allows testing of different of hardware platforms with only minor changes in coding.

The signal processing algorithms were developed in Ubuntu 14.04 and the GNU Radio SDK version is 3.7.7.1. The universal hardware driver (UHD) version used for the development is 003.008.004.

2.3 Adaptive Filter Algorithm

The adaptive algorithm that was used to change the FIR filter coefficients is the normalized least means squares (NLMS). NLMS is real-time recursive algorithm that adapts the FIR filter for every new sample received. The algorithm minimizes the mean square error of the difference between the noise received by the primary signal as shown on Fig. 1 and the reference noise.

The pseudocode for the normalized least means square algorithm implemented is shown below.

For each n

$$y(n) = \sum_{k=0}^n \mathbf{w}^H(k)x(n-k). \quad (1)$$

$$e(n) = d(n) - y(n) \quad (2)$$

$$\mathbf{w}_{j+1}(n+1) = \mathbf{w}_j(n) - \mu \frac{e^*n(n)}{\|x(n)\|^2 + \alpha} \quad (3)$$

where w is the FIR filter weight, x is the noise reference input, d is the primary input that contains the desired signal as well as the noise component, μ is the learning rate, and α is a small number that is greater than 0. α ensures that the algorithm is stable in the event $\|x(n)\|^2 = 0$. The algorithm was designed to have an α of 0.001, which guarantees stability but also minimizes its error contribution to the updated filter coefficient.

Equation 1 is the output of the FIR filter. The second equation finds the difference between the primary input and the output of FIR filter. If the output of the adaptive filter is equal to the noise component within the primary signal, then the error

should only contain the desired signal. If not, Eq. 3 constantly tunes the FIR filter weights at a rate of μ . The condition for μ must satisfy the following:

$$0 < \mu < 1$$

The learning rate can be arbitrarily chosen and largely depends on the type of environment in which the application is being used and what type of signals are involved. Theoretically, small values of μ increase the amount of time the algorithm takes to reach the optimal solution but produce fewer estimation errors. Larger values have the opposite effect and increase the stability of the algorithm.

The algorithm was developed in GNU Radio using C++. The code is available in the Appendix.

3 Performance

3.1 Test Setup

The performance of the interference cancellation system was evaluated in a closed-loop coaxial cable environment for initial testing. The diagram of the test setup is shown in Fig. 3.

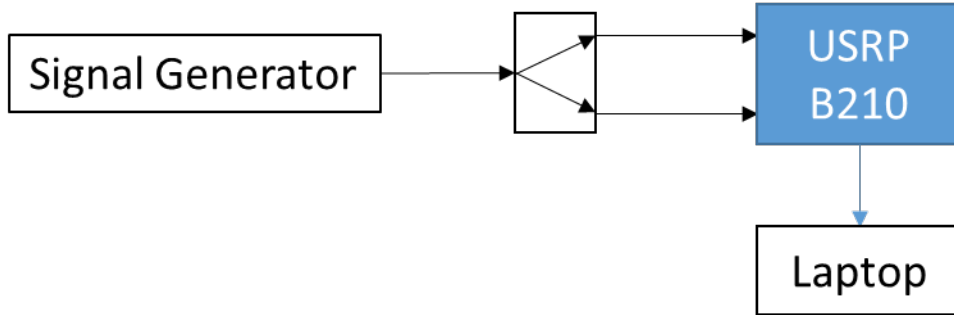


Fig. 3 Interference cancellation test setup

The signal generator generates a single tone signal that is fed into an RF splitter. The outputs of the RF splitter is then connected to each of the receive inputs of the USRP B210. The USRP B210 feeds all the samples to the laptop into the GNU Radio framework for signal processing. The laptop also shows the performance of the adaptive filter signal processing block written by plotting the Fourier transform in real time. The spectrum plots were captured to measure cancellation performance. The interference canceller was tested for various filter lengths from 1, 2, 3, and 5.

The splitter, Mini-Circuits ZFRSC-42-S+, and cable path incurs a loss of 7 dB at 300 MHz. When the B210 is configured in MIMO configuration, the B210 incurs another 8 dB of loss.

3.2 Test Parameters

The following test settings were used to configure the signal generator and the interference canceller.

Signal Generator:

- 1) Frequency: 300 MHz
- 2) Signal Type: Single Tone
- 3) Modulation Off
- 4) RF Amplitude: -60 dBm

GNU Radio Settings:

Adaptive Filter Parameters

- 1) Sampling Rate: 64 kHz
- 2) Filter Length: 1, 2, 3, and 5
- 3) Learning Rate $\mu = 0.01$

UHD Source

- 1) Sampling Rate: 64 kHz
- 2) Center Frequency: 300 MHz
- 3) Clock Frequency: 30.72 MHz

3.3 Test Results

The amount of cancellation performance was recorded by measuring the spectrum at the output of the adaptive filter. The measurements were taken at various filter lengths of 1, 2, 3, and 5. As shown in Figs. 4, 5, 6, and 7, the best cancellation performance achieved is when the adaptive filter length is 1. Figure 4 shows that the cancellation performance is achieved is 120 dB. The result is somewhat misleading because the noise floor of the B210 is at -125 dBm, and therefore, the realistic cancellation performance is $-60 - 8 - 7 - (-125) = 50 \text{ dB}$. The output of the filter also has a flat spectrum, indicating that the filter does not generate errors in performance. Filter lengths 2, 3, and 5 have noise suppression

Approved for public release; distribution unlimited.

levels of 40 dB at the center frequency but generate an additional frequency response that increases the noise in other bands. The amount of noise generated can be as large 15 dB above the noise floor.

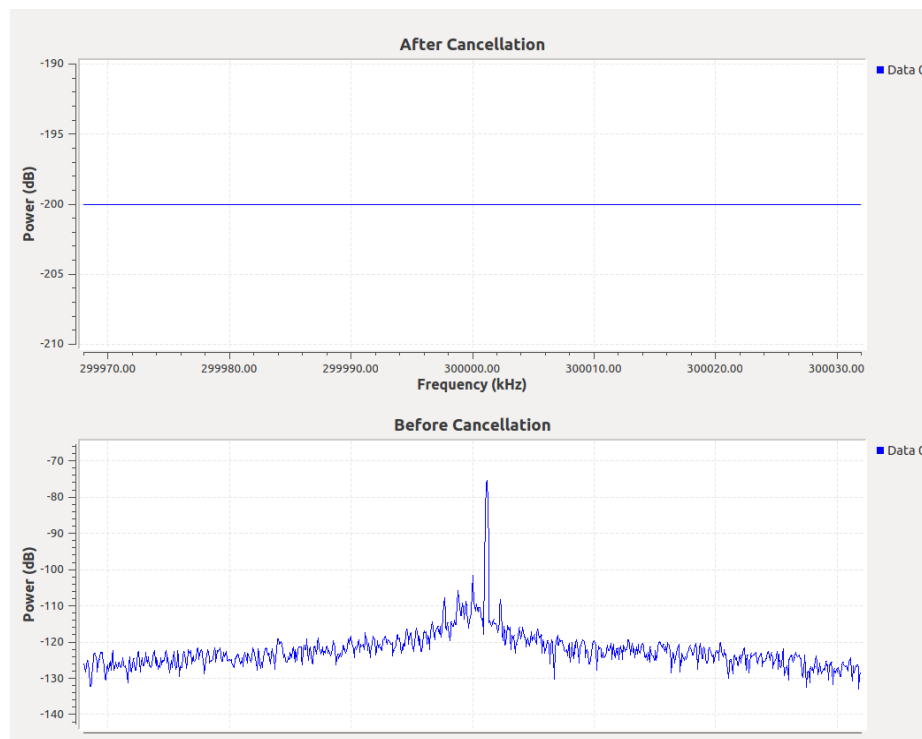


Fig. 4 Adaptive filter length of 1

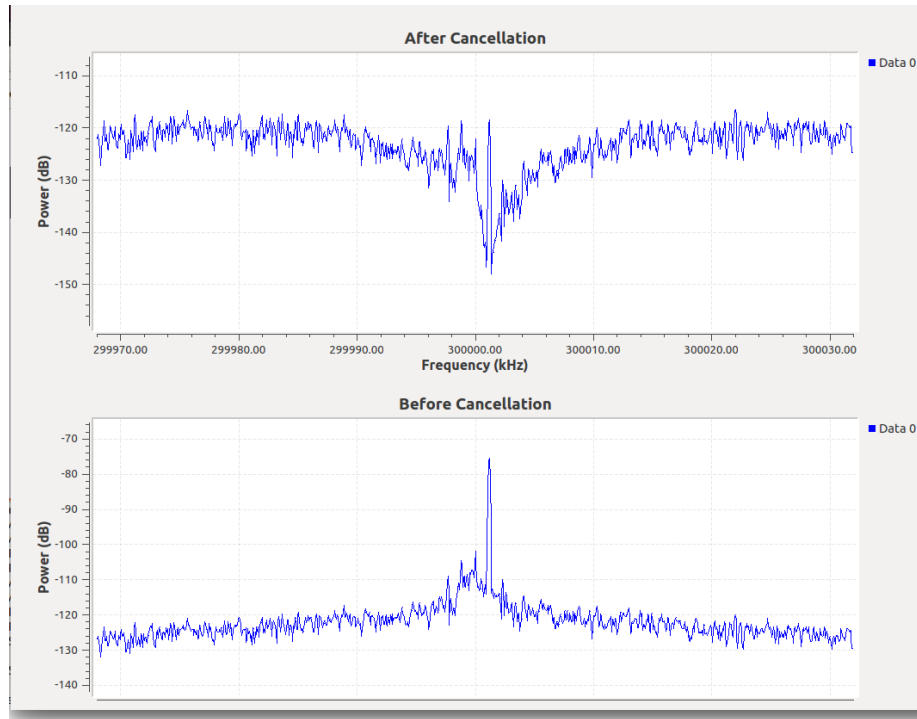


Fig. 5 Adaptive filter length of 2



Fig. 6 Adaptive filter length of 3

Approved for public release; distribution unlimited.

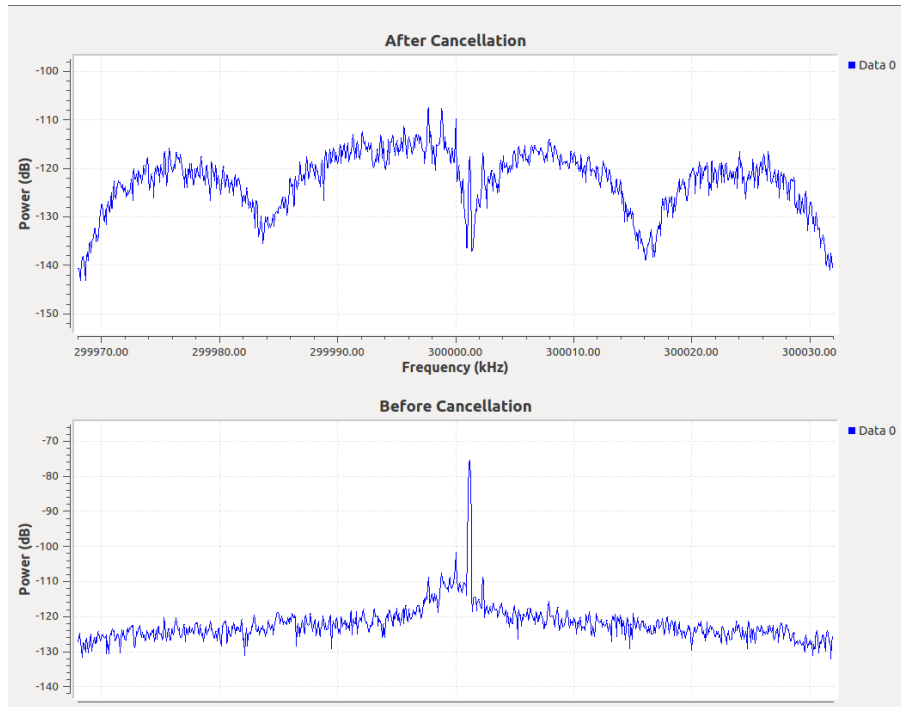


Fig. 7 Adaptive filter length of 5

The results of the performance of the interference canceller is not surprising. The filter length of 1 performs best because the test setup is a closed-loop setup with cables. The adaptive filter weights represent the impulse response of the test setup. In a closed-loop setup, the signal does not generate any delayed reflections that lengthen the impulse response of the propagation path. It is also possible that the reflections generated are below the noise floor of the radio and cannot be represented by the limited dynamic range of the radio. When the filter length is greater than 1 in a closed-loop system, the adaptive filter generates estimation errors that increase the noise at the output.

4. Summary and Conclusion

The performance of an interference cancellation prototype developed using GNU Radio and the Ettus radio B210 has been measured. The system is able to achieve cancellation levels up to 50 dB when the filter length is properly chosen.

The performance of the system can be further improved by using more expensive models from Ettus Research. The biggest limitation on cancellation performance ADCs is the B210. Other Ettus radios such as the N210 and the X310 have high performance 14-bit ADCs and a greater dynamic range.

5. References

Rondeau TW. On the GNU Radio Ecosystem. in Opportunistic Spectrum Sharing and White Space Access: The Practical Reality. Holland O, Bogucka H, Medeisis A., Ed. New York: Wiley, 2015.

Haykin SS. Adaptive Filter Theory. Englewood Cliff's, NJ: Prentice Hall, second ed., 1991.

INTENTIONALLY LEFT BLANK.

Appendix. GNURadio C++ Code

Approved for public release; distribution unlimited.

A-1 Adaptive Filter C++ Header File

```
#ifndef INCLUDED_ADAPTIVEFILTER_NLMS_IMPL_H
#define INCLUDED_ADAPTIVEFILTER_NLMS_IMPL_H

#include <AdaptiveFilter/nlms.h>
#include <gnuradio/filter/fir_filter.h>
#include <gnuradio/math.h>
#include <vector>
#include <stdexcept>
#include <gnuradio/gr_complex.h>

namespace gr {
    namespace AdaptiveFilter {

        class nlms_impl : public nlms, filter::kernel::fir_filter_ccc
        {
        private:
            gr_complex d_error;
            gr_complex d_FilterOutput;
            gr_complex d_norm;

            std::vector<gr_complex> d_new_taps;
            float d_mu;
            float d_FilterLen;

            bool d_updated;

        protected:
            gr_complex error(const gr_complex &in);
            void update_tap(gr_complex &tap, const gr_complex &in);

        public:
            nlms_impl(float mu, int FilterLen);
            ~nlms_impl();

            void set_taps(const std::vector<gr_complex> &taps);
            std::vector<gr_complex> taps() const;

            float gain() const
            {
                return d_mu;
            }
        }
    }
}
```

```

void set_gain(float mu)
{
    if(mu < 0.0f || mu > 1.0f) {
        throw std::out_of_range("NLMS::set_gain: Gain value must be in
[0,1]");
    }
    d_mu = mu;
}

// Where all the action really happens
int work(int noutput_items,
        gr_vector_const_void_star &input_items,
        gr_vector_void_star &output_items);
};

} // namespace AdaptiveFilter
} // namespace gr

#endif /* INCLUDED_ADAPTIVEFILTER_NLMS_IMPL_H */

```

A-2 Adaptive Filter C++ Source File

```
#ifndef HAVE_CONFIG_H
#include "config.h"
#endif

#include <gnuradio/io_signature.h>
#include "nlms_impl.h"

namespace gr {
  namespace AdaptiveFilter {

    using namespace filter::kernel;

    nlms::sptr
    nlms::make(float mu, int FilterLen)
    {
      return gnuradio::get_initial_sptr
        (new nlms_impl(mu, FilterLen));
    }

    /*
     * The private constructor
     */
    nlms_impl::nlms_impl(float mu, int FilterLen)
      : gr::sync_block("nlms",
        gr::io_signature::make(2, 2, sizeof(gr_complex)),
        gr::io_signature::make(1, 1, sizeof(gr_complex))),
        fir_filter_ccc(1, std::vector<gr_complex>(FilterLen, gr_complex(0,0))),
        d_new_taps(FilterLen, gr_complex(0,0)),
        d_updated(false), d_error(gr_complex(0,0))
    {
      set_gain(mu);
      if (FilterLen > 0){
        d_taps[0] = 1.0;
        d_new_taps[0] = 1.0;
      }
      fir_filter_ccc::set_taps(d_new_taps);
      set_history(FilterLen);
    }

    /*
     * Our virtual destructor.
     */
    nlms_impl::~nlms_impl()
    {
  
```

Approved for public release; distribution unlimited.

```

}
void
nlms_impl::set_taps(const std::vector<gr_complex> &taps)
{
    d_new_taps = taps;
    d_updated = true;
}

std::vector<gr_complex>
nlms_impl::taps() const
{
    return d_taps;
}

void
nlms_impl::update_tap(gr_complex &tap, const gr_complex &in)
{
    tap = std::conj(tap);
    tap += d_mu*std::conj(d_error)*in/(d_norm+.0001f);
}

gr_complex
nlms_impl::error(const gr_complex &sig)
{
    gr_complex error = sig - d_FilterOutput;
    return error;
}

int
nlms_impl::work(int noutput_items,
                gr_vector_const_void_star &input_items,
                gr_vector_void_star &output_items)
{
    const gr_complex *ref = (const gr_complex *) input_items[0];
    const gr_complex *sig = (const gr_complex *) input_items[1];
    gr_complex *out = (gr_complex *) output_items[0];

    if(d_updated) {
        d_taps = d_new_taps;
        set_history(d_taps.size());
        d_updated = false;
        return 0; // history requirements may have changed.
    }

    size_t k, l = d_taps.size();

```

Approved for public release; distribution unlimited.

```

size_t m = d_new_taps.size();

for ( int i = 0; i < noutput_items; i++) {
    d_norm = gr_complex(0.0f,0.0f);
    d_FilterOutput = gr_complex(0.0f,0.0f);
    d_FilterOutput = filter(&ref[i]);

    d_error = error(sig[i]);
    //Calculate the norm
    for( k = 0; k < l; k++) {
        d_norm += ref[i - k]*std::conj(ref[i - k]);
    }

    d_error = error(ref[i]);
    out[i] = d_error;

    for(k = 0; k < l; k++) {
        // Update tap locally from error.
        update_tap(d_taps[k], ref[i-k]);
        // Update aligned taps in filter object.
        fir_filter_ccc::update_tap(std::conj(d_taps[k]), k);
    }
}

return noutput_items;
}

} /* namespace AdaptiveFilter */
} /* namespace gr */

```

List of Symbols, Abbreviations, and Acronyms

ADC	analog-to-digital converter
ARL	US Army Research Laboratory
COTS	commercial-off-the-shelf
DAC	digital-to-analog converter
FIR	finite impulse response
FPGA	field-programmable gate array
MIMO	multiple input multiple output
NLMS	normalized least means squares
RF	radio frequency
SDK	software development kit
SDR	software-defined radio
UHD	universal hardware driver
USRP	universal software radio peripheral

1 DEFENSE TECHNICAL
(PDF) INFORMATION CTR
DTIC OCA

2 DIRECTOR
(PDF) US ARMY RESEARCH LAB
RDRL CIO LL
IMAL HRA MAIL & RECORDS MGMT

1 GOVT PRINTG OFC
(PDF) A MALHOTRA

2 DIRECTOR
(PDF) US ARMY RESEARCH LAB
RDRL SER U
J ACOSTA
A SULLIVAN